Parallel Computation of Three-Dimensional Flows using Overlapping Grids with Adaptive Mesh Refinement

Bill Henshaw

Centre for Applied Scientific Computing,

Lawrence Livermore National Laboratory,

Livermore, CA USA 94551

Don Schwendeman

Department of Mathematical Sciences,

Rensselaer Polytechnic Institute, Troy, NY 12180

SIAM Conference on Parallel Processing for Scientfic Computing,

Atlanta, GA March 2008.

Background: Schlieren image from a detonation hitting a collection of moving rigid cylinders.

Acknowledgments

Supported by

Department of Energy, Office of Science

MICS Program: Mathematical, Information, and Computational Sciences

SciDAC: Scientific Discovery through Advanced Computing

Current Overture developers (`www.llnl.gov/casc/Overture`)

Kyle Chand

Bill Henshaw

Introduction:

We are interesting in numerically solving well-posed initial-boundary-value problems

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} = \mathcal{L}(\mathbf{u},\mathbf{x},t), & t>0, \quad \mathbf{x}\in\Omega, \\ \mathbf{u}(\mathbf{x},t) = \mathbf{u}_0(\mathbf{x}), & t=0, \quad \mathbf{x}\in\Omega, \\ \mathcal{B}(\mathbf{u},\mathbf{x},t) = 0, & t>0, \quad \mathbf{x}\in\partial\Omega. \end{cases}$$

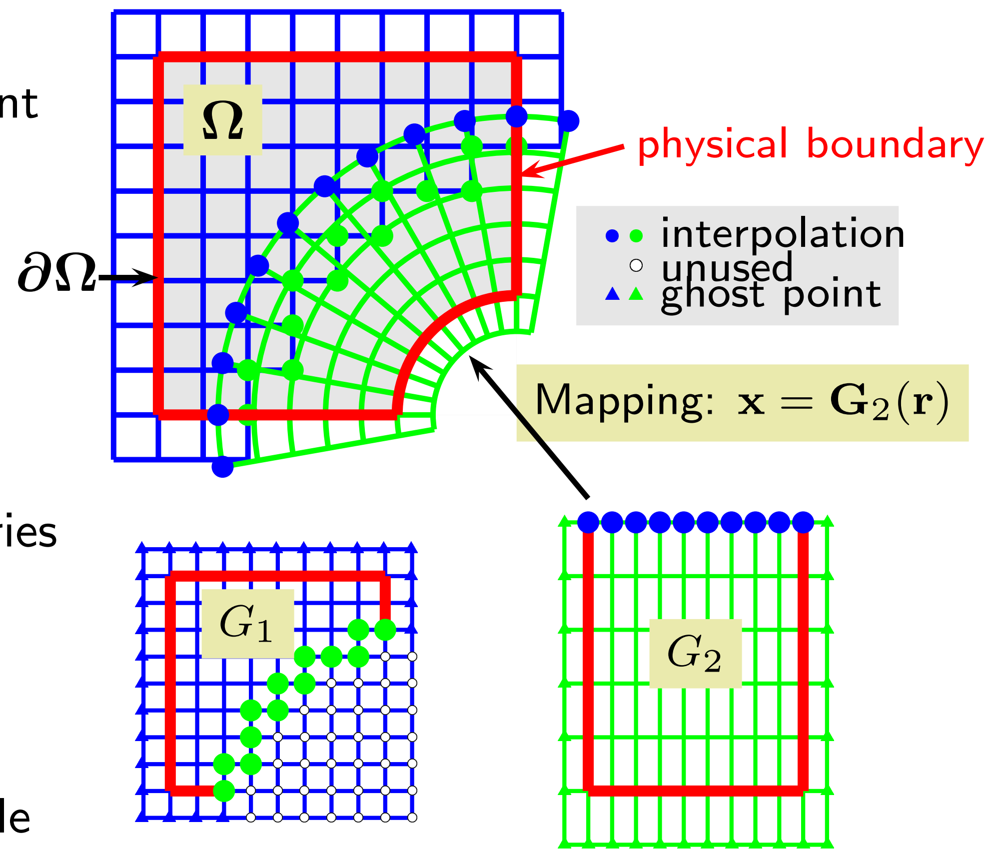
in complex three-dimensional domains $\Omega \in \mathbb{R}^3$.

$\diamondsuit$ We use overlapping (overset/Chimera) grids to discretize the domain Ω and finite-difference or finite-volume methods to approximate the PDE.

$\diamondsuit$ If the solutions exhibit localized multiscale behaviour such as sharp fronts, interfaces, shocks, reaction zones etc. then the use of adaptive mesh refinement (AMR) can reduce the time-to-solution or allow higher-resolution results for given computational resources.

Background: Overlapping grids for solving Partial Differential Equations

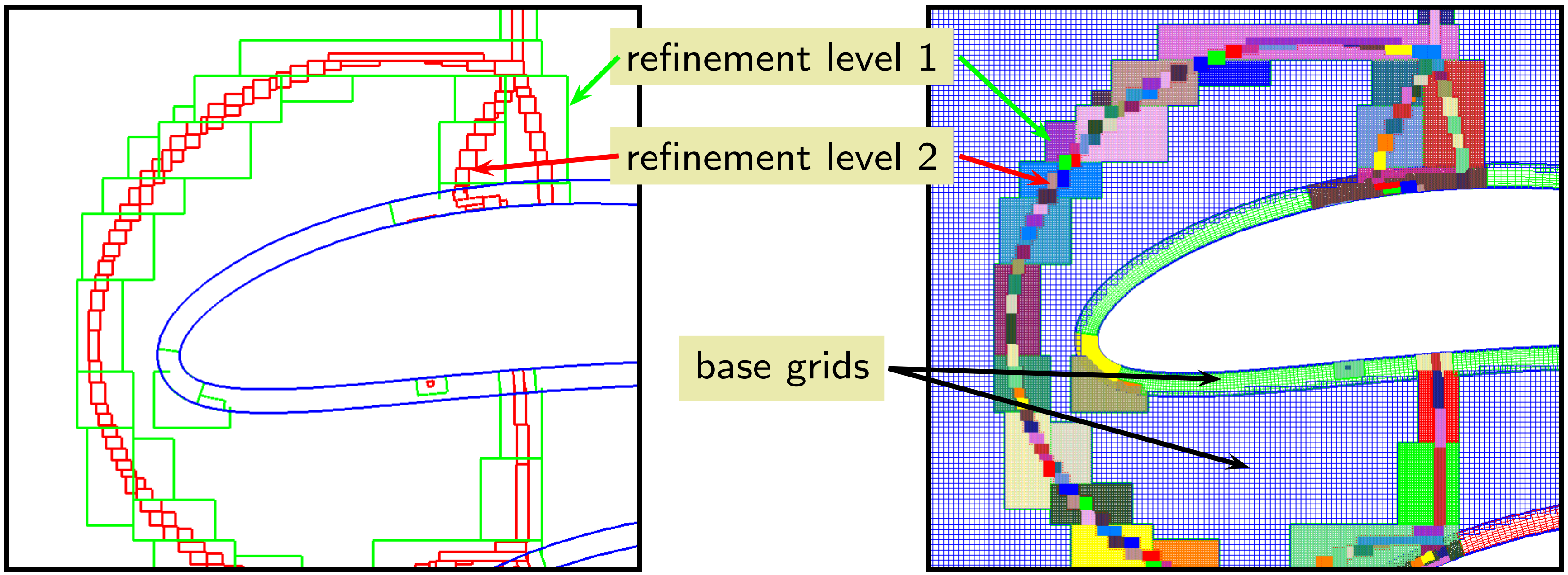
- A set of structured grids that overlap.
- Solutions matched by interpolation.
- Body fitted grids permit accurate treatment of boundary conditions.
- A grid generator (Ogen) is used to automatically connect component grids, but component grid generation is not yet fully automatic.
- Grids can be rapidly generated as boundaries move.
- Efficient high-order accurate methods are possible.
- Algorithms must take into account multiple grids and interpolation points.



Claim: If designed properly, an algorithm for overlapping grids can be asymptotically as fast and memory efficient as an algorithm for a single Cartesian grid.

Block Structured Adaptive Mesh Refinement and Overlapping Grids

- Refinement patches are generated in the parameter space of each component grid (base grid).
- Refinement patches are organized in a hierarchy of *refinement levels*.
- Error estimators determine where refinement is needed.
- AMR grid generation (Berger-Rigoutsos algorithm) builds refinement patches based on the error estimate.
- refinement grids may interpolate from refinement grids of different base grids.
- The key issue is efficiency.



Parallel Adaptive Mesh Refinement on Overlapping Grids

We have recently developed the parallel capabilties for AMR on overlapping grids.

Parallel Issues:

♢ Overlapping grids: parallel grid generation for the initial grid; updating interpolation points on AMR grids, parallel interpolation.

♢ AMR: Error-estimation, regridding, interpolation.

♢ Parallel Distributions of Arrays and Load balancing

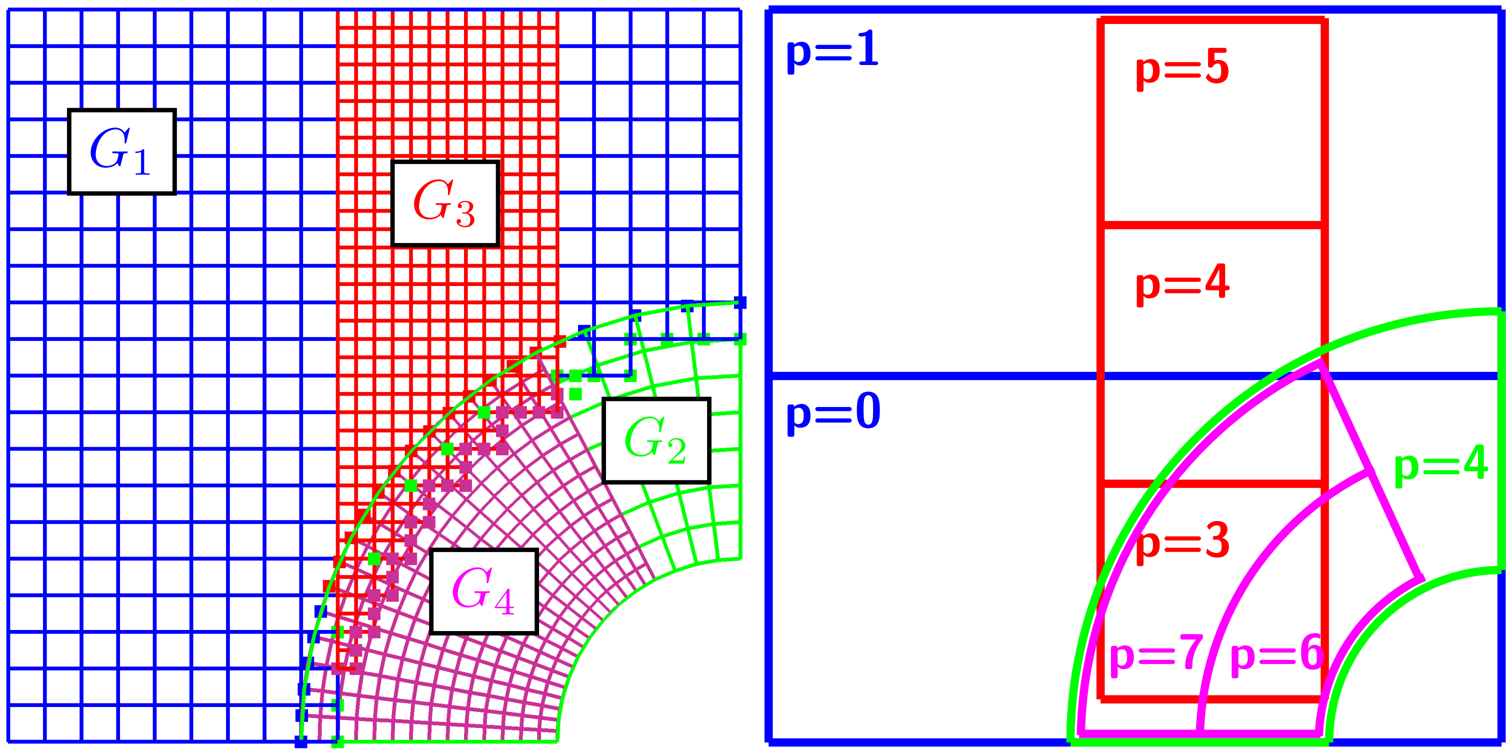
♢ I/O and Graphics

Reference:

WDH., D. W. Schwendeman, *Parallel Computation of Three-Dimensional Flows using Overlapping Grids with Adaptive Mesh Refinement*, UCRL-JRNL-236681, Submitted for publication, 2007.

WDH., D. W. Schwendeman, *Moving Overlapping Grids with Adaptive Mesh Refinement for High-Speed Reactive and Nonreactive Flow*, J. Comp. Phys. **216** (2005) 744-779.

Distributing Overlapping and AMR grids in Parallel



Each base grid or refinement grid can be distributed over a contiguous range of processors. In this example the base grid G_1 is distributed over processors $[0, 1]$, the base grid G_2 over processor $[4]$, the refinement grid G_3 over processors $[3, 4, 5]$ and the refinement grid G_4 over processors $[6, 7]$.

The AMR Time-Stepping Algorithm

PDEsolve$(\mathcal{G}, t_{\text{final}})$

$\{\ \ t := 0;\ n := 0;$

$\quad \mathbf{u}_{\mathbf{i}}^{n} := \mathbf{applyInitialCondition}(\mathcal{G});$

$\quad \underline{\mathbf{while}}\ t < t_{\text{final}}$

$\qquad \underline{\mathbf{if}}\ (n \ \mathbf{mod}\ n_{\text{regrid}} \equiv 0)$ *// rebuild the AMR grids*

$\qquad\quad e_{\mathbf{i}} := \mathbf{estimateError}(\mathcal{G}, \mathbf{u}_{\mathbf{i}}^{n});$

$\qquad\quad \mathcal{G}^{*} := \mathbf{regrid}(\mathcal{G}, e_{\mathbf{i}});$

$\qquad\quad \mathbf{u}_{\mathbf{i}}^{*} := \mathbf{interpolateToNewGrid}(\mathbf{u}_{\mathbf{i}}^{n}, \mathcal{G}, \mathcal{G}^{*});$

$\qquad\quad \mathcal{G} := \mathcal{G}^{*};\ \mathbf{u}_{\mathbf{i}}^{n} := \mathbf{u}_{\mathbf{i}}^{*};$

$\qquad\quad \underline{\mathbf{end\ if}}$

$\qquad \Delta t := \mathbf{computeTimeStep}(\mathcal{G}, \mathbf{u}_{\mathbf{i}}^{n});$

$\qquad \mathbf{u}_{\mathbf{i}}^{n+1} := \mathbf{advancePDE}(\mathcal{G}, \mathbf{u}_{\mathbf{i}}^{n}, \Delta t);$ *// take a time step*

$\qquad \mathbf{interpolate}(\mathcal{G}, \mathbf{u}_{\mathbf{i}}^{n+1});$ *// interpolate overlapping grid points*

$\qquad \mathbf{applyBoundaryConditions}(\mathcal{G}, \mathbf{u}_{\mathbf{i}}^{n+1}, t + \Delta t);$

$\qquad t := t + \Delta t;\quad n := n + 1;$

$\quad \}$

Error-Estimation

The error-estimator we generally use is

$$e_{\mathbf{i}} = \sum_{k=1}^{m} e_{k,\mathbf{i}} + \tau_{\mathbf{i}} \ , \tag{1}$$

where the error in solution component k (e.g. ρ,u,v,...) is estimated as weighted sum of first- and second-order undivided differences,

$$e_{k,\mathbf{i}} = \frac{1}{3} \sum_{\alpha=1}^{3} \left(\frac{c_1}{s_k} |\Delta_{0\alpha} U_{k,\mathbf{i}}| + \frac{c_2}{s_k} |\Delta_{+\alpha} \Delta_{-\alpha} U_{k,\mathbf{i}}| \right) . \tag{2}$$

When solving the reactive Euler equations, we add $\tau_{\mathbf{i}}$, which is an estimate of the truncation error in the sub-cycled chemistry terms.

$\diamondsuit$ The error-estimator is smoothed using a Jacobi iteration on the entire overlapping grid so that the error propagates onto neighbouring grids. Refinement grids thus include a *buffer region*.

AMR Regriding

♢ Every few times steps (e.g. every 8 times steps) the error is estimated and a new set of AMR grids are found. We use a modified Berger-Rigoutsos algorithm.

AMR Interpolation

♢ **AMR boundary-interpolation** : during each time step, ghost points on AMR grid boundaries are interpolated from grids at the same level or a coarser level.

♢ **Refinement grid transfer step** : when the locations of the AMR grids are recomputed, the solution values from the new grid-hierarchy are interpolated from the solution values on the old grid-hierarchy

Remark: Our first implementation of AMR regriding and interpolation has great room for improvement to reduce the number of messages being passed.

Load-Balancing

The aim of load-balancing is to distribute the computational work-load amongst the processors in a nearly even fashion.

Constraints:

♢ Each grid can be distributed across a contiguous range of processors (a constraint imposed by the version of Multiblock PARTI that we use).

The Algorithm is based on a best fit decreasing bin-packing algorithm:

♢ starting from the largest grid, split the grid into a number of regularly shaped pieces of some estimated optimal size.

♢ pack the pieces of the grid onto a contiguous set of processors. Go to the next largest grid and repeat.

♢ Check the final load balance. If poorly balanced, repeat the process but split the grids into more pieces.

Load-Balancing

Notes:

◇ The target load balance can always be achieved by splitting each grid across all processors.

◇ Communication costs are not explicitly taken into account.

◇ Any variation in computational cost per grid point is currently not taken into account.

Validation: Solving an advection-diffusion problem with parallel AMR

We consider the solution of the initial-boundary-value problem for the advection-diffusion equation:

$$
\begin{cases}
\dfrac{\partial u}{\partial t} + \mathbf{a}\cdot\nabla u = \nu\Delta u + f(\mathbf{x},t), & t>0, \quad \mathbf{x}\in\Omega, \\
u = u_0(\mathbf{x}), & t=0, \quad \mathbf{x}\in\Omega, \\
u = g(\mathbf{x},t), & t>0, \quad \mathbf{x}\in\partial\Omega,
\end{cases}
$$

where $u = u(\mathbf{x},t)$ is a scalar function, $\mathbf{a} = \mathbf{a}(\mathbf{x},t) \in \mathbb{R}^3$ is a given velocity, $\nu > 0$ is a constant diffusivity and $f(\mathbf{x},t)$ is a given forcing function.

These are discretized on curvilinear grids using the mapping-method, resulting in the system of ODEs,

$$
\frac{d}{dt}U_{\mathbf{i}}(t) + \mathbf{a}\cdot\nabla_h U_{\mathbf{i}}(t) = \nu\Delta_h U_{\mathbf{i}}(t) + f_{\mathbf{i}}(t),
$$

These equations are advanced in time using a second or fourth-order Runge-Kutta method, RK2 or RK4.

The Method of Analytic Solutions (aka Twilight-zone flow)

For the advection-diffusion IBVP

$$
\begin{cases}
\dfrac{\partial u}{\partial t} + \mathbf{a}\cdot\nabla u = \nu\Delta u + f(\mathbf{x},t), & t>0, \quad \mathbf{x}\in\Omega, \\
u = u_0(\mathbf{x}), & t=0, \quad \mathbf{x}\in\Omega, \\
u = g(\mathbf{x},t), & t>0, \quad \mathbf{x}\in\partial\Omega,
\end{cases}
$$

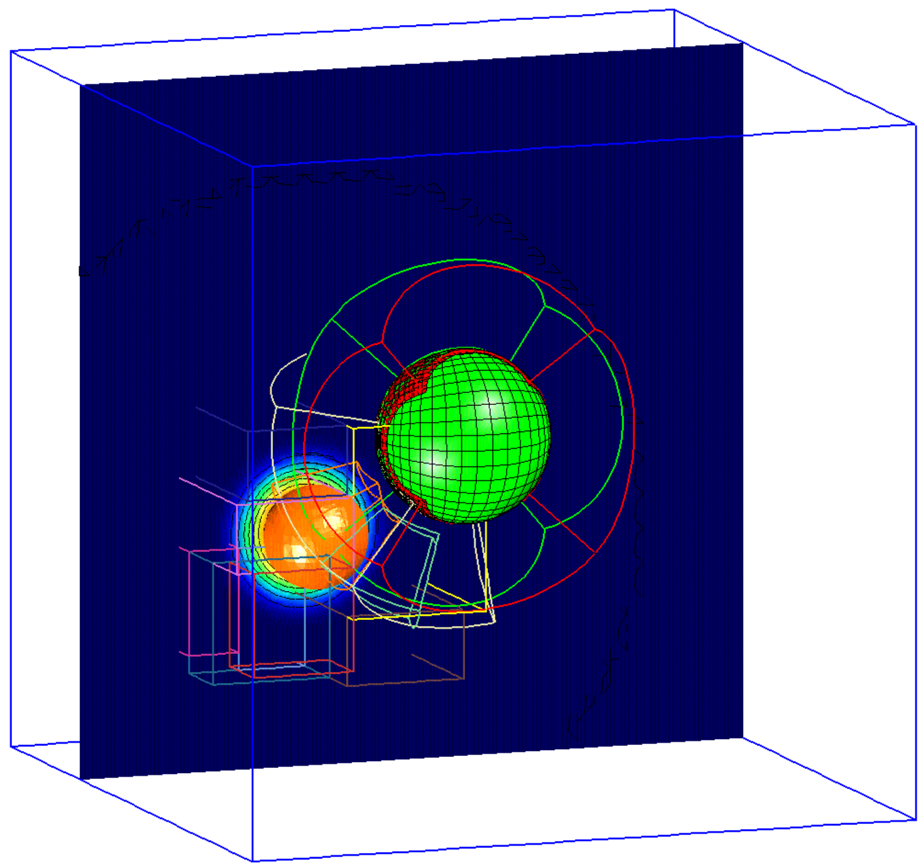
We can make any smooth function $\bar{u}(\mathbf{x},t)$ an exact solution by choosing

$$
\begin{aligned}
& f(\mathbf{x},t) = \bar{u}_t + \mathbf{a}\cdot\nabla\bar{u} - \nu\Delta\bar{u}, \qquad u_0(\mathbf{x}) = \bar{u}(\mathbf{x},0), \\
& g(\mathbf{x},t) = \bar{u}(\mathbf{x},t), \qquad \text{for } \mathbf{x}\in\partial\Omega.
\end{aligned}
$$

We often choose $\bar{u}(\mathbf{x},t)$ to be a low degree polynomial since our approximations are often exact in this case on Cartesian grids.

A good exact-solution for tesing AMR is the translating pulse

$$\bar{u}(\mathbf{x},t) = c_0 \exp\left\{-\left(|\mathbf{x}-\mathbf{x}_c(t)|/c_1\right)^2\right\},$$
$$\mathbf{x}_c(t) = \mathbf{x}_0 + \mathbf{v}_0 t.$$



Above: a pulse moving through a sphere-in-a-box grid. Refinement grid boxes are shown.

Advection-diffusion: moving pulse in a sphere-in-a-box

Notation: $\mathcal{G}_{\mathrm{s}}^{(j,l)}$: j: base grid resolution factor, l: number of additional refinement levels.
n_r : refinement ratio, $\mathcal{E}_{j,1}$: maximum error.

Grid	n_r	N_{proc}	N_{step}	N_{regrid}	$\mathcal{N}_{\mathrm{grid}}$	$\mathcal{N}_{\mathrm{point}}$	$\mathcal{E}_{j,1}$
$\mathcal{G}_{\mathrm{s}}^{(1,1)}$	2	32	48	24	(3, 23)	2.0e+5	2.84e−2
$\mathcal{G}_{\mathrm{s}}^{(2,1)}$	2	32	120	60	(3, 49)	1.1e+6	6.91e−3
$\mathcal{G}_{\mathrm{s}}^{(3,1)}$	2	32	376	188	(3, 128)	6.7e+6	1.70e−3

Parallel AMR results for runs involving the sphere-in-a-box grid with the moving pulse solution. Convergence rate $\sigma = 2.0$ (second-order accurate)

Grid	n_r	N_{proc}	N_{step}	N_{regrid}	$\mathcal{N}_{\mathrm{grid}}$	$\mathcal{N}_{\mathrm{point}}$	$\mathcal{E}_{j,\ell}$
$\mathcal{G}_{\mathrm{s}}^{(1,2)}$	2	8	126	64	(13, 53)	6.0e+5	7.25e−3
$\mathcal{G}_{\mathrm{s}}^{(1,2)}$	2	32	126	64	(13, 53)	6.0e+5	7.25e−3
$\mathcal{G}_{\mathrm{s}}^{(1,1)}$	4	16	187	47	(3, 21)	6.6e+5	7.25e−3
$\mathcal{G}_{\mathrm{s}}^{(1,1)}$	4	32	187	47	(3, 21)	6.6e+5	7.25e−3
$\mathcal{G}_{\mathrm{s}}^{(2,1)}$	2	1	120	60	(3, 49)	1.1e+6	6.91e−3
$\mathcal{G}_{\mathrm{s}}^{(2,1)}$	2	32	120	60	(3, 49)	1.1e+6	6.91e−3
$\mathcal{G}_{\mathrm{s}}^{(4,0)}$	–	8	166	–	(3, 3)	4.9e+6	6.76e−3
$\mathcal{G}_{\mathrm{s}}^{(4,0)}$	–	32	166	–	(3, 3)	4.9e+6	6.76e−3

The effective resolution is the same for all runs and we observe that the numerical errors, $\mathcal{E}_{j,\ell}$, are approximately equal.

Solving the reactive Euler equations.

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial x_1}\mathbf{F}_1(\mathbf{u}) + \frac{\partial}{\partial x_2}\mathbf{F}_2(\mathbf{u}) + \frac{\partial}{\partial x_3}\mathbf{F}_3(\mathbf{u}) = \mathbf{H}(\mathbf{u}),$$

where

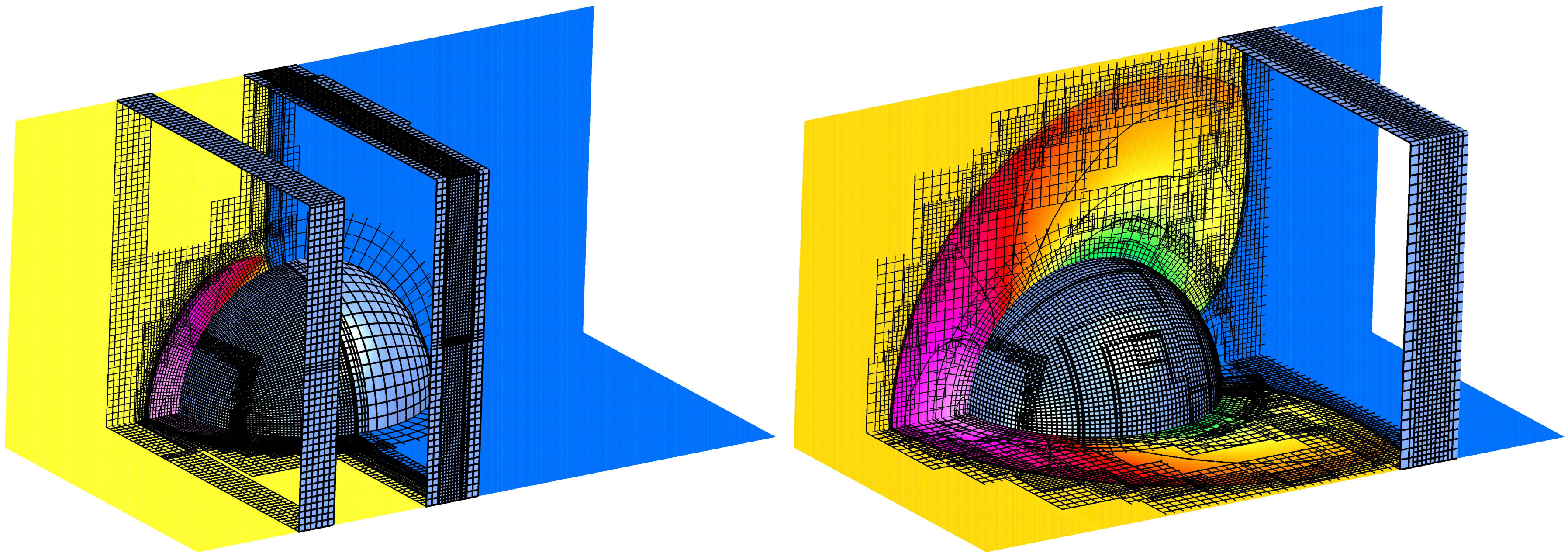
$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ E \\ \rho\mathbf{s} \end{bmatrix}, \qquad \mathbf{F}_n = \begin{bmatrix} \rho v_n \\ \rho v_n \mathbf{v} + p\mathbf{e}_n \\ v_n(E+p) \\ \rho v_n \mathbf{s} \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \\ \rho\mathbf{R} \end{bmatrix}.$$

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\mathbf{v}|^2 + \rho q,$$

$\diamondsuit$ The numerical approximation uses a second-order extension of Godunov's method.

$\diamondsuit$ The stiff source term in the reactive case is handled using a Runge-Kutta error-control scheme.

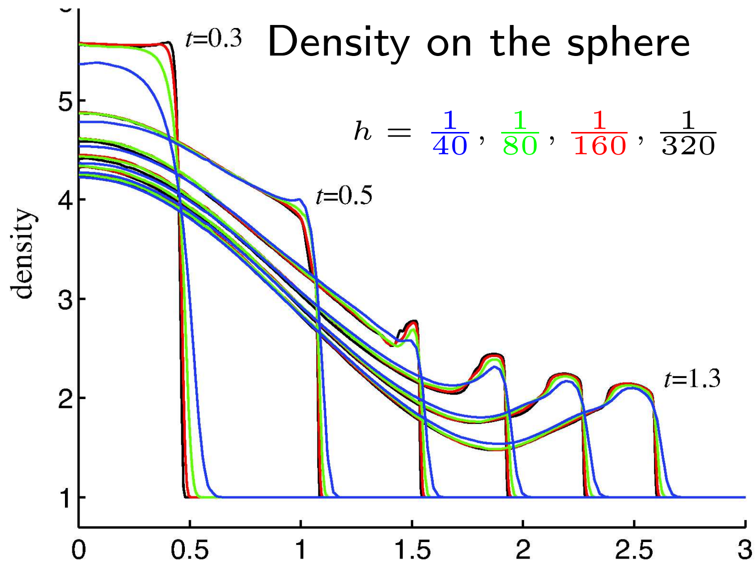
AMR grids for shock diffraction by a quarter sphere



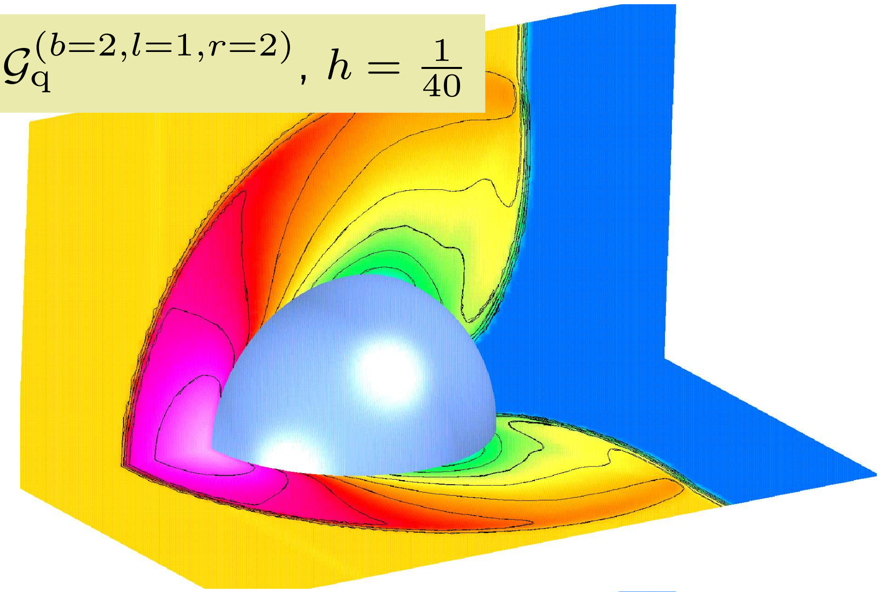
Density and AMR grids for the quarter-sphere problem at $t = 0.6$ (left) and $t = 1.4$ (right). (The grid is coarsened by a factor of 4 for illustrative purposes.)

Notes: Euler equations computed with cgcns: two-levels of refinement factor 2, 32 processors, from 6 to 1827 grids, a maximum of 55 million grid points.

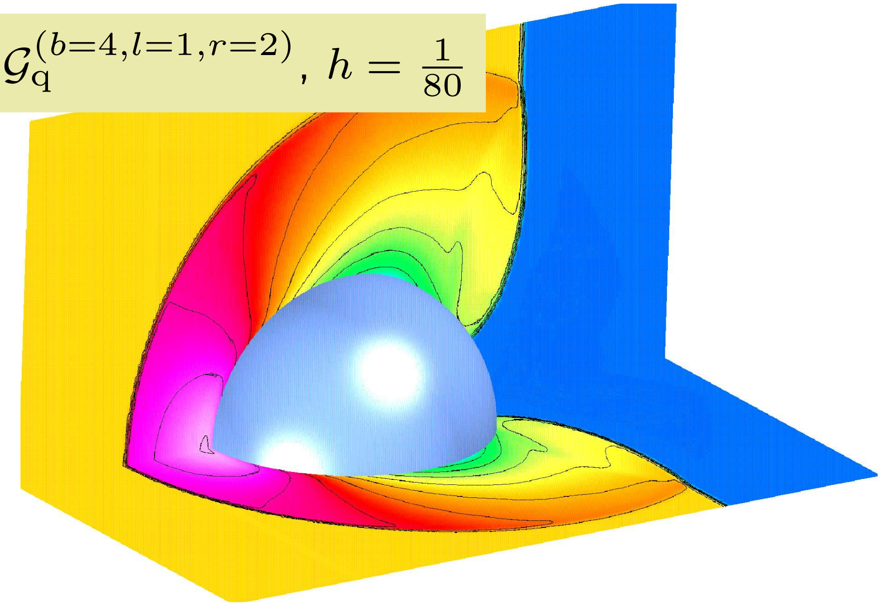
Grid convergence study for shock diffraction by a quarter sphere



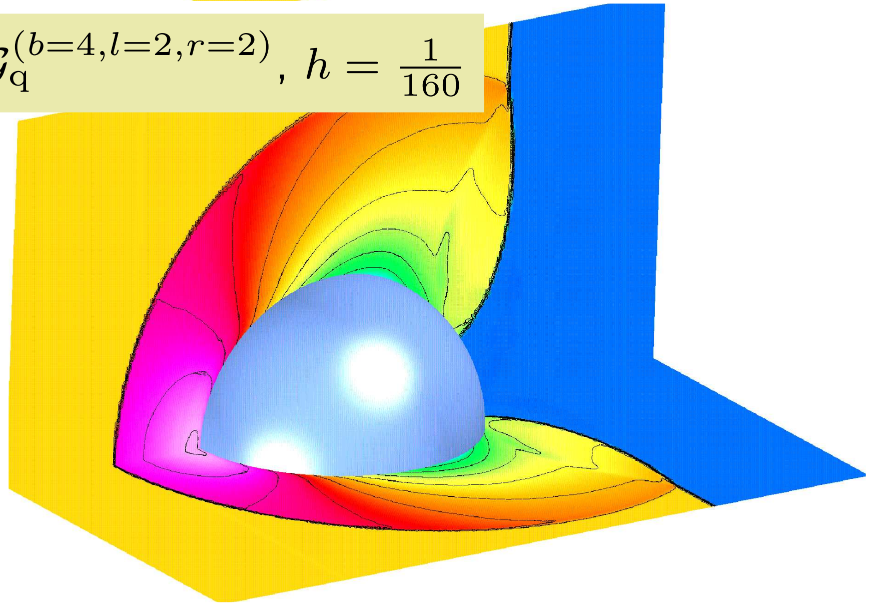
$\mathcal{G}_{\rm q}^{(b=2,l=1,r=2)}$, $h = \frac{1}{40}$



$\mathcal{G}_{\rm q}^{(b=4,l=1,r=2)}$, $h = \frac{1}{80}$



$\mathcal{G}_{\rm q}^{(b=4,l=2,r=2)}$, $h = \frac{1}{160}$



Parallel AMR, shock diffraction by a sphere - strong scaling results

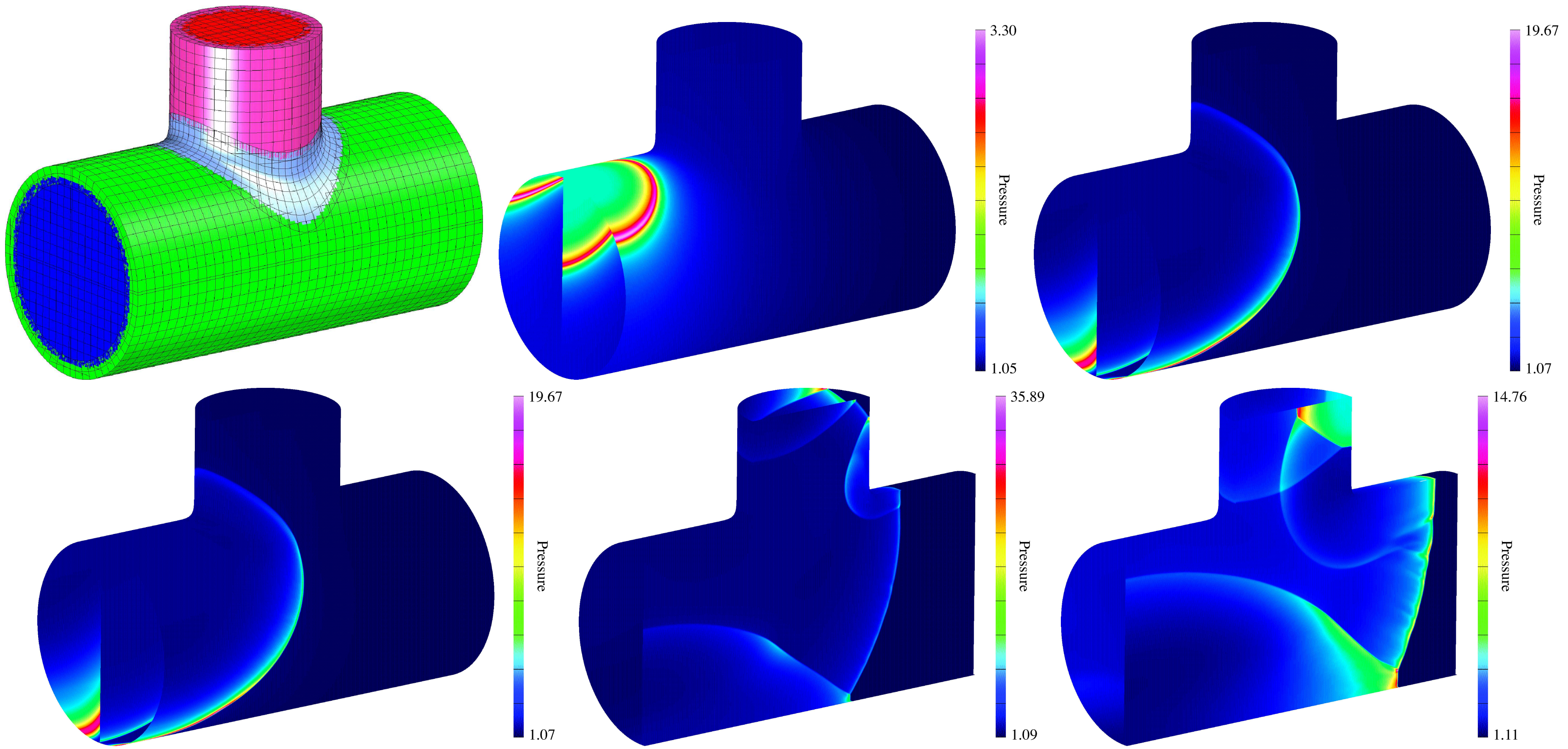
k	Grid	$\mathcal{N}_{\rm point}^{(k)}$	$N_{\rm proc}^{(k)}$	$\mathcal{N}_{\rm point}^{(k)}/N_{\rm proc}^{(k)}$	$N_{\rm step}^{(k)}$	$\mathcal{T}_k$	$\mathcal{S}_k$
0	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	1	2.01e+6	617	15.2	1.00
1	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	2	1.00e+6	617	7.77	0.98
2	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	4	5.02e+5	617	3.96	0.96
3	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	8	2.51e+5	617	2.09	0.91
4	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	16	1.26e+5	617	1.09	0.87
5	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	32	6.27e+4	617	0.587	0.81
6	$\mathcal{G}_{\rm q}^{(4,0)}$	2.01e+6	64	3.14e+4	617	0.341	0.70

Strong scaling results with no AMR. $\mathcal{T}_k =$ CPU time in seconds per step. The parallel scaling factor $\mathcal{S}_k$ should be 1 for perfect parallel scaling.

k	Grid	$\mathcal{N}_{\rm point}^{(k)}$	$N_{\rm proc}^{(k)}$	$\mathcal{N}_{\rm point}^{(k)}/N_{\rm proc}^{(k)}$	$N_{\rm step}^{(k)}$	$\mathcal{T}_k$	$\mathcal{S}_k$
0	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	1	1.61e+6	645	11.8	1.00
1	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	2	8.05e+5	645	6.23	0.95
2	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	4	4.02e+5	645	3.23	0.91
3	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	8	2.01e+5	645	1.82	0.81
4	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	16	1.01e+5	645	1.02	0.72
5	$\mathcal{G}_{\rm q}^{(2,1)}$	1.61e+6	32	5.03e+4	645	0.591	0.62

Strong scaling results with AMR. The current parallel AMR interpolation functions send too many small messages; these need to be merged.

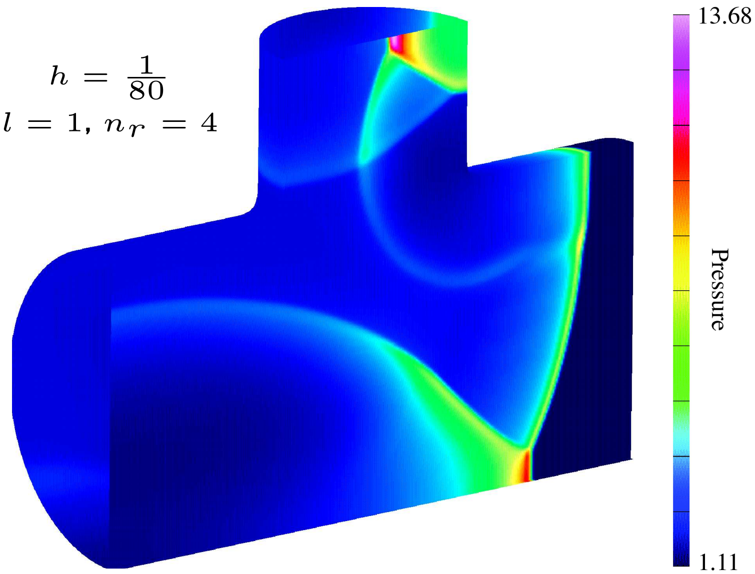
Cgcns parallel AMR example: detonation initiation in a T-shaped pipe



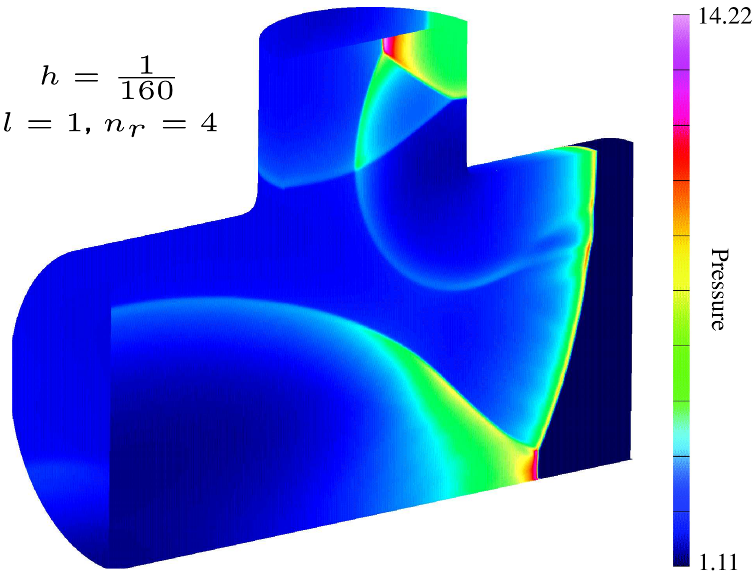
Notes: Reactive-Euler equations computed with cgcns: one level of refinement factor 4, 4930 time steps, 48 processors, from 5 to 682 grids, a maximum of 100 million grid points (effective resolution of 400 million).

Grid convergence study for a detonation in a T-pipe

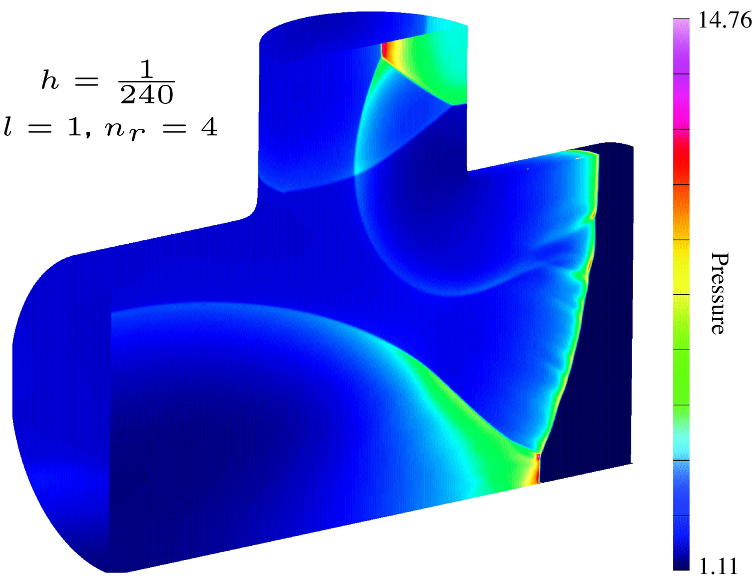
$h = \frac{1}{80}$
$l = 1,\ n_r = 4$



$h = \frac{1}{160}$
$l = 1,\ n_r = 4$



$h = \frac{1}{240}$
$l = 1,\ n_r = 4$



Summary

- We have developed an approach for solving time dependent PDEs using overlapping grids and AMR on parallel, distributed-memory computers.
- Each base grid or refinement grid can be independently distributed across one or more processors. A modified bin-packing algorithm is used as the load-balancer.
- The accuracy of the approach was validated by solving advection-diffusion equation with the method of analytic solutions.
- The approach was further validated by solving the Euler-equations and reactive Euler-equations.
- The method showed reasonably good parallel scaling up to 64 processors. Further work is required to the initial implementation to reduce communication costs.
- Future work: moving grids and AMR in parallel.